# A project-based immersion system

Vincent Ribaud, Philippe Saliou

# A project-based immersion system

Vincent Ribaud and Philippe Saliou
*Département informatique, Université de Brest, C.S. 93837, 29238 Brest Cedex 3*
*{Vincent.Ribaud, Philippe.Saliou}@univ-brest.fr*

***Abstract***

*We present features of an education system entirely based on a 7-months project, performed by a 6-students team within a virtual company and tutored by an experimented software engineer. We describe some aspects of a past project: functions of the system, technology, deliverables, and assessment. Students' outcomes are drafted in term of roles.*

## 1. Introduction

Recognizing a core body of knowledge is pivotal to the development and accreditation of university curricula and the licensing and certification of professionals. The Software Engineering 2004 Volume [1] provides two major outcomes:

- SEEK: Software Engineering Education Knowledge - what every SE graduate must know
- Curriculum: ways that this knowledge and the skills fundamental to software engineering can be taught in various contexts

In the SE2004 volume, Guidelines for SE Curriculum Design and Delivery emphasize that how SEEK topics should be taught may be as important as what is taught. The Guideline 14 states that the curriculum should have a significant real-world basis.

Several analyses of software engineering teaching emphasise the advantages of a long-term team project (one semester or a year) [2]. The studio is the central training method in architecture schools and this analogy was used to provide a suitable educational environment for software design [6]. Students work in teams on a large-scale project supervised by faculty members, and generally for an external client. Behind software studios is Donald Schön's idea of the reflective practitioner perspective [5]. The generalized educational setting is a reflective practicum, where students learn mainly by doing, with the help of coaching.

This paper briefly presents an education system designed from this perspective and shows an example of a project performed.

## 2. Apprenticeship by immersion

### 2.1. Overview

Since 2002, the curriculum of the last year of Brest Masters' in Software Engineering is entirely based on a real-world basis, which is a 7-months project, performed by 6-students team within a virtual company and tutored by an experimented software engineer. We call this education system "Software engineering apprenticeship by immersion" [4].

The learning process is achieved in two iterations. During the first iteration (4.5 months), students are swapped around the different tasks needed by engineering activities and strongly guided by the tutor. During the second iteration (2 months), roles are fixed within each team, and teams are relatively autonomous in completing the project, with the tutor fulfilling a role that is mainly a supervising and rescuing activity.

We designed and built the immersion system with two major objectives in mind: to guide and accompany learners as they build knowledge and skills - the learning process - and to provide an educational environment in which realistic working situations are experienced - the reflective practicum.

This paper will focus on two of the six essential elements of the immersion system: a breakdown of apprenticeships into three SE processes subdivided in SE activities, and a set of apprenticeships scenes providing the learning environment and defining tasks; a project environment reproducing the context of a Management Information System project.

### 2.2. Software engineering process

We use a hierarchical process/activity/scenes model (adapted from the ISO/IEC 12207) as a reference framework [3]. From the 25 processes of ISO/IEC 12207, we concentrate on those related to software development cycle, that is: 5.3 Development, 6.1 Documentation, 6.2 Configuration Management, 6.3 Quality Assurance, 6.4 Verification, 6.5 Validation, 7.1 Management, and 7.2 Infrastructure. We reorganized the selected processes in 3 processes: Development Engineering, Project Management and Development Support.

Roughly speaking, an apprenticeship scene is intended to 'set to music' the learning goals of an apprenticeship situation related to a software engineering activity. A scene involves roles, activities and resources. Several scenes take place simultaneously. During the same period of time, students work in subgroups on different activities belonging to different processes. Furthermore, the learning objectives of an activity could require the cut out in different scenes closely coupled.

The complete cycle of scenes is temporally organized into stages. Each stage (from 1 to 3 weeks) groups several scenes and carries on activities belonging to the three processes. The cycle of the first iteration is: Stage 0 : Introduction (1.5 week) ; Stage 1 : Project and means set-up (1 week) ; Stage 2 : Requirement capture - Architecture choices (2.5 weeks) ; Stage 3 : Requirements consolidation (2.5 weeks) ; Stage 4 : Analysis - Technical exploration (3 weeks) ; Stage 5 : Design tailoring (1.5 weeks) ; Stage 6 : Design (1.5 week ) ; Stage 7 : Realization - Integration (2.5 weeks) ; Stage 8 : Validation (1 week) ; Stage 9 : Deployment - Verification (1 week).
With this work model and the progress reality, the tutor defines the set of tasks at the beginning of the stage and prepares apprenticeship cards describing the scenes of the stage. Then, the tutor's assigns tasks to students and supervises progress.

### 2.3. The immersion environment

"Studios are typically organized around manageable projects of design, individually or collectively undertaken, more or less closely patterned on projects drawn from actual practice" [6]. The immersion environment extends the concept of development studio to a real-scale dimension and aims to achieve the same goals, at a minimum. Tomayko reports that "the use of a well-established development process, a matrix organization, and one-to-one mentoring give the highest return on investment" [7]. The immersion environment reproduces a software project environment in the field of the Management Information System:

- Dedicated rooms constitute the theatre of operations: a landscape room, with individual working post, for each team; a meeting room; an engine room.
- A software development process: the 2-Track Unified Process with a functional way and a technical architecture way.

- Thanks an agreement with Thales group, an ISO 9001 corporate baseline defining good practices and capitalizing the company's know-how.
- A working framework including common installations and tool suites intended for the manufacturing and the documentation of software products.

## 3. The eCompas project

During the academic year 2006-2007, the eCompas system was developed by a team of students from our immersion system.

### 3.1. Functions

The eCompas system is intended to help students, faculty and academic administrators to manage development, assessment and value-added competencies over the course of a curriculum. Four main functions have been defined, each of which can be helped by eCompas functionalities: these functions are: competency model management; personal follow-up of competencies; situation analysis of competencies, and institutional supervision.

### 3.2. Technical environment

The eCompas system uses a three-tier architecture in which the user interface, functional process logic, computer data storage and data access are developed and maintained as independent modules, on separate platforms. The Oracle Application Development Framework (Oracle ADF) was used to develop the system. Oracle ADF is an end-to-end application framework that builds on J2EE standards and open-source technologies. Oracle ADF achieve a clean separation of business logic, page navigation, and user interface by adhering to a model, view, controller (MVC) architecture.

### 3.3. Documentation

The tutor wrote the invitation to tender including the statement of work, a response to solicitation with a technical offer answering the expected needs.

Main deliverables provided by the students are: Meeting report, Project Plan, Requirement Specification, Software Analysis, Software Design, Code, Integration and Validation Plan, Software User Manual, and Software Operator Manual.

Besides these engineering documents, students produce other kinds of document related to their apprenticeship: case study, usage guide, evaluation report, book or article summary, best practices, etc.

### 3.4. Assessment

When a product is delivered, the tutor carefully examines it and writes an assessment card including a general assessment together with all the points to be improved or to start over. The feedback is given in front of the authors, which allows authors to delve deeper, discuss, and even contest remarks made by the tutors. Following this briefing, students have to update or start their product again, after which it will be assessed again. Generally, two assessment iterations are necessary in order to guarantee a satisfactory result – or at least sufficient to carry on with the project.

## 4. Learning outcomes

The SE 2004 Volume provides a generic list of students' outcomes. Graduates of our system are able to achieve - at least partially - the 7 cited outcomes. Rather than discuss each outcome, we prefer to present an overall view of students' outcomes in terms of roles.

First of all, students learn to live together. Each student lives with their team mates for a period of 6 months, 5 days a week, 8 hours a day. They learn respect for others, for their work, for their own workplace and the shared infrastructure.

Students' main role is to build the project. The student is in turn architect, project manager, manufacturer, inventor, and artist. He/she must understand that his/her own work takes place within a structured set and that the success of each piece is necessary to the success of the project as a whole. The understanding of long-term issues should always be kept in mind, and regular and sustained effort is essential.

Alongside this building activity, the apprentice engineer must sometimes transform him/herself into an explorer. Students cannot always find their way through the maze of educational resources provided, or in the explanations they can get from tutors. So there is nothing to help them through the work they have to do. This situation is aimed at pushing students to deepen existing knowledge, discover new skills, and invent personal solutions.

Each of these three previous roles brings us back to the question of teamwork and hence to the definition of what 'a team member' is. Students become aware of the fact that nothing can be done without others - or more precisely, that they need the others to do everything.

## 5. Conclusion

The immersion system uses a process of learning and exploration that is central in a reflective practicum: within which experiential knowledge is built, through the practice of a reflexive analysis.

The relevance and resistance to change of our education system probably comes from the fact that each viewpoint can be linked to an activity breakdown at two or three levels of the envisaged software engineering profession.

## 6. References


[1] ACM and IEEE, Software Engineering 2004, http://sites.computer.org/ccse (last accessed February 13th, 2008).

[2] B. Meyer, "Software Engineering in the Academy", IEEE Computer Volume 34 Issue 5, IEEE Computer Society Press, May 2001, pp. 28-35.

[3] ISO/IEC 12207:1995, Information technology -- Software life cycle processes, International Organization for Standardization (ISO), Geneva, Switzerland.

[4] V. Ribaud, and Ph. Saliou, "Software Engineering Apprenticeship by Immersion", International Workshop on Patterns in Teaching Software Development, ECOOP 2003, LNCS Volume 3013/2004 ,Germany, 2003, p. 137

[5] D. Schön, The reflective practitioner:, Basic Books, New York, 1983.

[6] J. E. Tomayko, "Carnegie Mellon's software development studio: a five year retrospective", In Proceedings of the 9th Conference on Software Engineering Education, IEEE Computer Society Press, pp. 119-129.